# MPI/OPENMP AFFINITY PERFORMANCE MAPPING AND TESTING

Timothy H. Kaiser, Ph.D.
tkaiser2@nrel.gov

NREL
Transforming ENERGY

# NATIONAL RENEWABLE ENERGY LABORATORY

## NREL Mission and Vision

NREL strives to achieve our vision of a clean energy future for the world through our mission: leading research, innovation, and strategic partnerships to deliver solutions for a clean energy economy.

## About High-Performance Computing at NREL

NREL's high-performance computing (HPC) system users tap into the largest HPC environment in the world dedicated to advancing renewable energy and energy efficiency technologies.

https://www.nrel.gov/

NREL Transforming ENERGY

# ABSTRACT

The National Renewable Energy Lab has just launched (and submitted a top 500 Run) its newest HPC platform, Kestrel, with 2144 nodes with dual Intel Sapphire Rapids processors.

We'll discuss MPI/OpenMP affinity mapping and testing. We'll present a hybrid MPI/OpenMP test code that reports affinity as a function of environmental settings, tasks, and threads. We show that without attentiveness to affinity, performance can be adversely effected. However, we'll show how to get ideal mapping, where tasks and threads are laid out for performance.

We present a batch script that can be run to sweep over various command line, environmental settings and task/thread combinations. In addition to Intel compilers the script will test Cray, MPICH, and OpenMPI.

We recommend the test code be run before a production run to ensure the desired mappings of tasks and threads. A git repository will be available with all codes and scripts.

# REPO

https://github.com/timkphd/affinity.git

# NREL'S KESTREL ENVIRONMENT

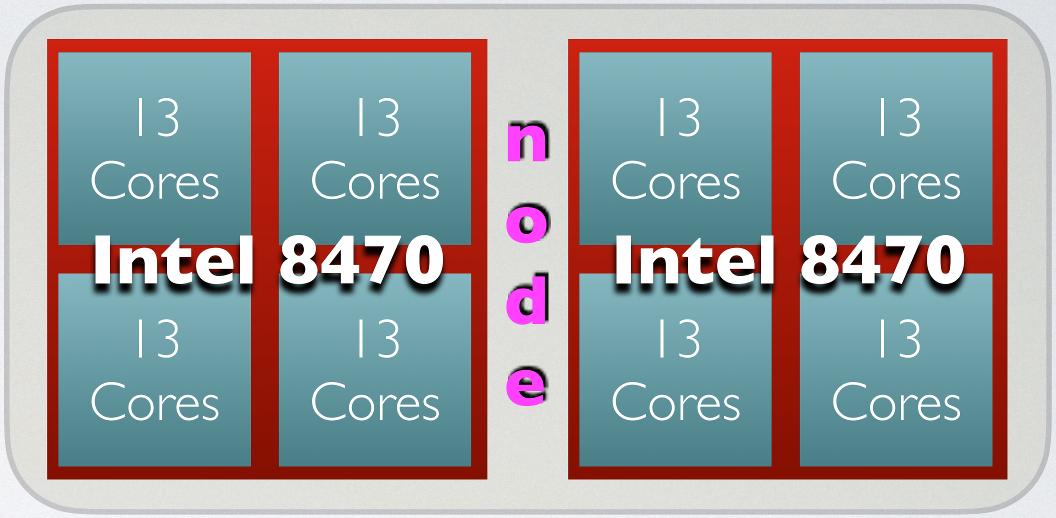
- Slurm
- HPE (Cray) with Slingshot
  - Cray Programming environment - supported by a module system
    - Cray MPI (mpich based)
      - Cray Fortran and C
      - GCC
      - Intel ifort and icc
  - Intel MPI with ifort and icc
  - Others also, but we'll skip for today

# CURRENT KESTREL CONFIGURATION

| Number of Nodes | Processors | Memory | Accelerators | Local Storage |
|---|---|---|---|---|
| 2144 | Dual socket **4th Gen Intel® Xeon® Scalable Processors (52-core)** | 256 GB DDR5 | N/A | 256 nodes with 1.92 TB NVMe M.2 |
| 10 | Dual socket **4th Gen Intel® Xeon® Scalable Processors (52-core)** | 2 TB DDR5 | N/A | 8 x 1.6 TB NVMe |
| 8 | Dual socket **4th Gen Intel® Xeon® Scalable Processors (52-core)** | 256 GB DDR5 | 2 NVIDIA A40 GPUs | 2 x 3.84 TB NVMe |

# AFFINITY & WHY IMPORTANT

- Affinity - mapping of threads/tasks to cores
- Kestrel 104 cores/node
    - 2 chips (Intel 8470)
        - 52 cores each
        - 4 "tiles" with 13 cores each




- **Worst case: Multiple threads/tasks can end up on the same core potentially reducing performance by 2X or maybe much more**
- Also: You may want to put threads/tasks on particular tiles to maximize communications or memory access
- Possible to have different MPI tasks to have different # threads

# OUR EXAMPLES

- **phostone.c fhostone.F90**
  - Hello world on steroids
  - **Hybrid MPI / OpenMP**
  - Many command line options
  - Will use options to print out a line for each MPI task and OpenMP thread along with the node and core on which it is running: **Show affinity**
  - Will run for 7 seconds.

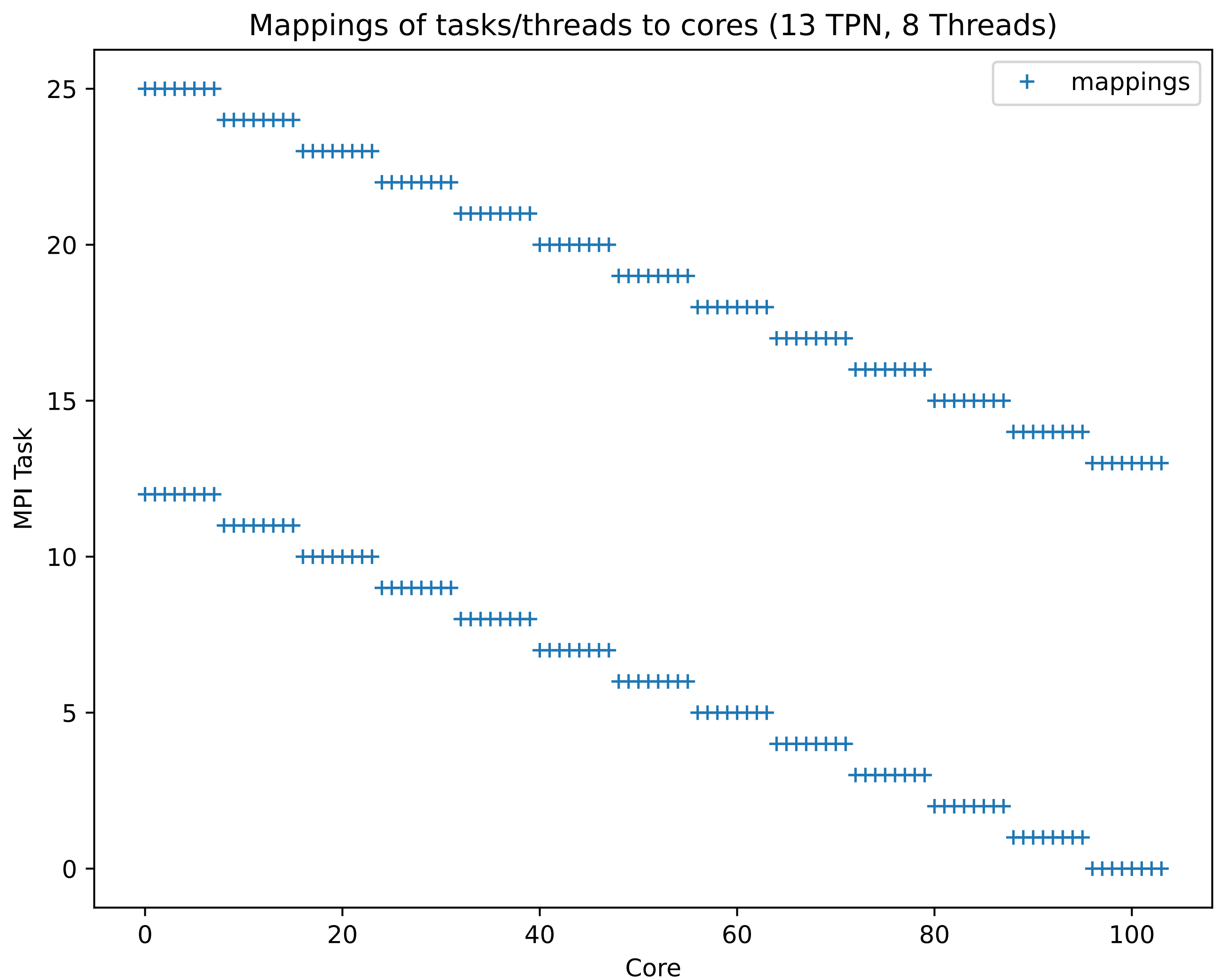
# PHOSTONE.C OUTPUT

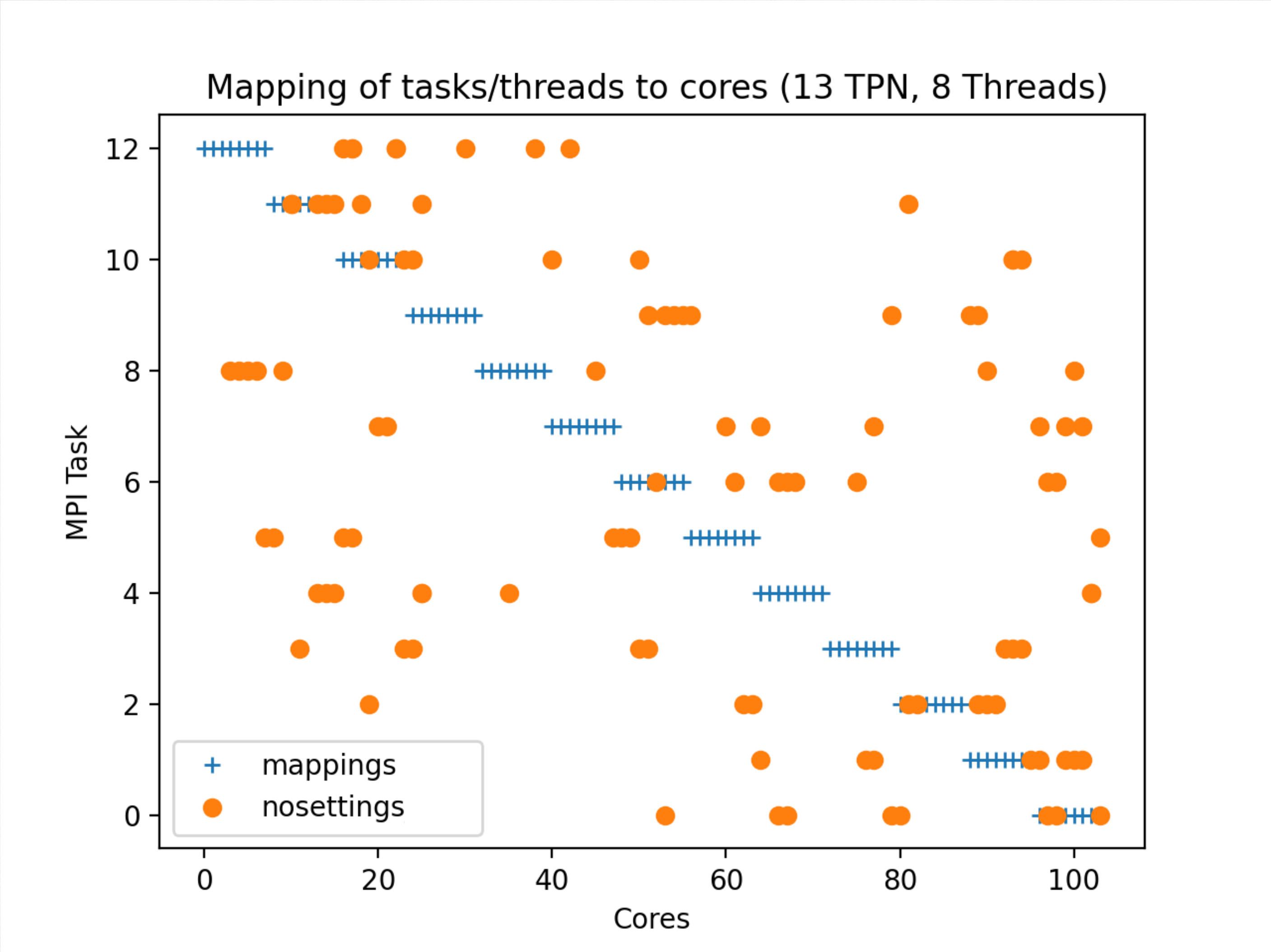
```
MPI VERSION Intel(R) MPI Library 2021.10 for Linux* OS
task     thread             node name  first task    # on node  core
0000      0000         X1005C4S5B0N0         0000         0000  0000
0000      0001         X1005C4S5B0N0         0000         0000  0001
0000      0002         X1005C4S5B0N0         0000         0000  0002
0000      0003         X1005C4S5B0N0         0000         0000  0003
0000      0004         X1005C4S5B0N0         0000         0000  0004
0000      0005         X1005C4S5B0N0         0000         0000  0005
0000      0006         X1005C4S5B0N0         0000         0000  0006
0000      0007         X1005C4S5B0N0         0000         0000  0007

0001      0000         X1005C4S5B0N0         0000         0001  0052
0001      0001         X1005C4S5B0N0         0000         0001  0053
0001      0002         X1005C4S5B0N0         0000         0001  0054
0001      0003         X1005C4S5B0N0         0000         0001  0055
0001      0004         X1005C4S5B0N0         0000         0001  0056
0001      0005         X1005C4S5B0N0         0000         0001  0057
0001      0006         X1005C4S5B0N0         0000         0001  0058
0001      0007         X1005C4S5B0N0         0000         0001  0059

0002      0000         X1005c4s6n0n0         0002         0000  0000
0002      0001         X1005c4s6n0n0         0002         0000  0001
0002      0002         X1005c4s6n0n0         0002         0000  0002
0002      0003         X1005c4s6n0n0         0002         0000  0003
0002      0004         X1005c4s6n0n0         0002         0000  0004
0002      0005         X1005c4s6n0n0         0002         0000  0005
0002      0006         X1005c4s6n0n0         0002         0000  0006
0002      0007         X1005c4s6n0n0         0002         0000  0007

0003      0000         X1005c4s6n0n0         0002         0001  0052
0003      0001         X1005c4s6n0n0         0002         0001  0053
0003      0002         X1005c4s6n0n0         0002         0001  0054
0003      0003         X1005c4s6n0n0         0002         0001  0055
0003      0004         X1005c4s6n0n0         0002         0001  0056
0003      0005         X1005c4s6n0n0         0002         0001  0057
0003      0006         X1005c4s6n0n0         0002         0001  0058
0003      0007         X1005c4s6n0n0         0002         0001  0059
total time      7.001
```

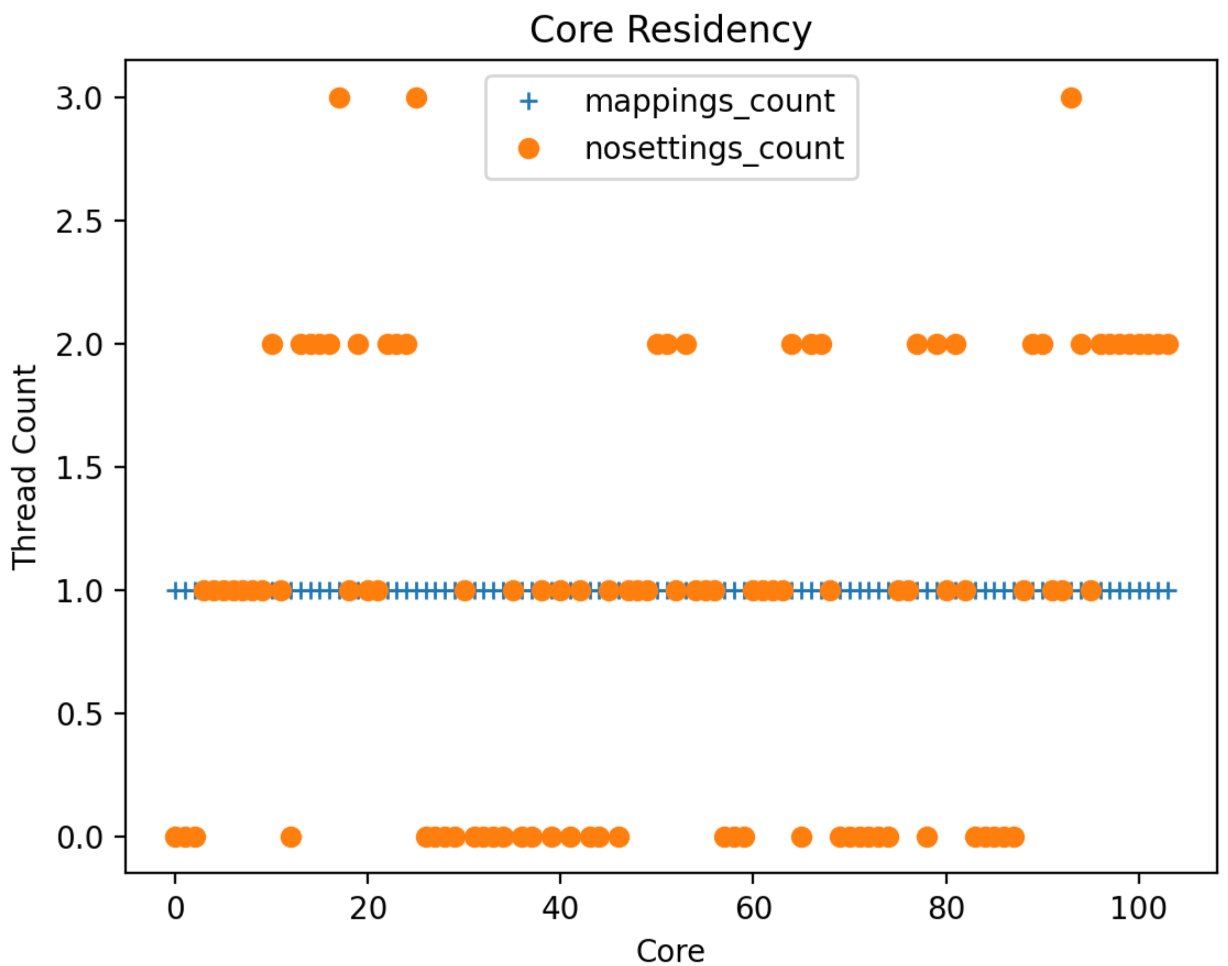
- Intel MPI
- 2 nodes
- 2 MPI tasks / node
- 8 OpenMP threads
- Sorted and enhanced

# RUN COMMANDS

- Since phostone and fhostone are hybrid MPI/OpenMP codes we will set OpenMP variables.
  - # threads
  - # thread binding
- We'll also add some options to the srun line to tests/ensure we are getting good mapping of tasks and threads to cores
- We can run with various numbers of MPI tasks per node and number of OpenMP threads per task but the two multiplied together should not exceed then number of cores on a node (104).


Mappings of tasks/threads to cores (13 TPN, 8 Threads)
mappings
MPI Task
Core
0
5
10
15
20
25
0
20
40
60
80
100


Mapping of tasks/threads to cores (13 TPN, 8 Threads)
MPI Task
Cores
0
2
4
6
8
10
12
0
20
40
60
80
100
mappings
nosettings


Core Residency
mappings_count
nosettings_count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Thread Count
0
20
40
60
80
100
Core

# OUR SCRIPT...

- Creates a new directory with "everything"
- Makes all versions of code
- Loops over:
  - Various tasks per node / Threads per task
  - Environment settings and srun command line options
    - Each Compiler/MPI combinations

# OUR EXAMPLE SCRIPT

- Reports lots of information
  - Bindings for each run
  - Normal program output for phostrun  includes mapping of tasks and threads to nodes and cores
  - MPI launch times
- Final output is a report of successful/failed mapping
  - Success = expected unique combinations of nodes and cores
  - Good News: It works for all tested versions of MPI and mappings (with the correct settings)

NREL Transforming ENERGY

# CUT TO THE CHASE:

- With the proper setting in sbatch scripts and the srun command we are able to "trivially" get apps to behave reasonably for all cases I tested on Kestrel, Swift and Eagle.
- Masks (a mapping list) allow a fine grain placement of tasks & threads to cores

# OVERKILL FOR MOST PEOPLE

- While you can run this for the full set you might not want to use the allocation hours (minutes)
- Suggested use...
  - Run phostone using the exact run arguments you use for your production code to see how it maps tasks and threads to cores

# PrgEnv-*

PrgEnv-amd/8.3.3
PrgEnv-aocc/8.3.3
PrgEnv-cray-amd/8.3.3
PrgEnv-cray/8.3.3
PrgEnv-gnu-amd/8.3.3
PrgEnv-gnu/8.3.3
PrgEnv-intel/8.3.3
PrgEnv-nvhpc/8.3.3
PrgEnv-nvidia/8.3.3

- Modules for using Cray's MPI with various backend compilers
- Red one currently work (cray, gnu, intel)
- Blue ones are for AMD processors and Nvidia GPUs (coming)
- PrgEnv-cray/8.3.3 is the default
- **All of these use the same MPI Library (Cray-MPICH)**

NREL Transforming ENERGY

# "PURE" INTEL SUITE

```
module purge
module load intel-oneapi
module load intel-oneapi-mpi
module load gcc/13.1.0
```

# Example Output Intel MPI

```
MPI VERSION Intel(R) MPI Library 2021.10 for Linux* OS

task     thread             node name  first task    # on node  core
0000       0000          X1001C3S0B1N1        0000         0000  0007
0000       0002          X1001C3S0B1N1        0000         0000  0003
0000       0003          X1001C3S0B1N1        0000         0000  0004
0000       0007          X1001C3S0B1N1        0000         0000  0000
0000       0006          X1001C3S0B1N1        0000         0000  0006
0000       0005          X1001C3S0B1N1        0000         0000  0001
0000       0001          X1001C3S0B1N1        0000         0000  0002
0000       0004          X1001C3S0B1N1        0000         0000  0005
0001       0000          X1006C6S0B0N0        0001         0000  0007
0001       0006          X1006C6S0B0N0        0001         0000  0004
0001       0002          X1006C6S0B0N0        0001         0000  0002
0001       0007          X1006C6S0B0N0        0001         0000  0005
0001       0004          X1006C6S0B0N0        0001         0000  0006
0001       0003          X1006C6S0B0N0        0001         0000  0000
0001       0005          X1006C6S0B0N0        0001         0000  0003
0001       0001          X1006C6S0B0N0        0001         0000  0001
mpi_init 0 1698853027.3182 1698853027.8130       0.4949
mpi_init 1 1698853027.3013 1698853027.8136       0.5124
total time      7.004
```

# Example Output PrgEnv-*

```
MPI VERSION MPI VERSION    : CRAY MPICH version 8.1.23.5 (ANL base 3.4a2)
MPI BUILD INFO : Tue Nov 29 12:42 2022 (git hash 210ae8b)

task    thread             node name  first task    # on node  core
0000      0001         X1005C2S3B1N1        0000         0000  0001
0000      0003         X1005C2S3B1N1        0000         0000  0003
0000      0005         X1005C2S3B1N1        0000         0000  0005
0000      0004         X1005C2S3B1N1        0000         0000  0004
0000      0006         X1005C2S3B1N1        0000         0000  0006
0000      0000         X1005C2S3B1N1        0000         0000  0000
0000      0002         X1005C2S3B1N1        0000         0000  0002
0000      0007         X1005C2S3B1N1        0000         0000  0007
0001      0000         X1005C3S1B0N0        0001         0000  0000
0001      0002         X1005C3S1B0N0        0001         0000  0002
0001      0001         X1005C3S1B0N0        0001         0000  0001
0001      0006         X1005C3S1B0N0        0001         0000  0006
0001      0004         X1005C3S1B0N0        0001         0000  0004
0001      0005         X1005C3S1B0N0        0001         0000  0005
0001      0007         X1005C3S1B0N0        0001         0000  0007
0001      0003         X1005C3S1B0N0        0001         0000  0003
mpi_init 0 1696732984.4138 1696732984.5317       0.1179
mpi_init 1 1696732984.3726 1696732984.5323       0.1597
total time      7.004
```

# makefile

## Intel MPI

```
SHELL:=/usr/bin/bash

recurse:
	module purge                    ; \
	module load intel-oneapi        ; \
	module load intel-oneapi-mpi    ; \
	module load gcc/13.1.0          ; \
	$(MAKE) -f $(firstword $(MAKEFILE_LIST)) both

both: f.impi c.impi pp.impi

#defines USEFAST
include makefile.include

ifeq ($(USEFAST),yes)
OPS=-DUSEFAST
EXTRA=getcore.o
endif

F90=mpiifort
CC=mpiicc

f.impi: fhostone.F90 $(EXTRA)
	$(F90) $(OPS) $(EXTRA) -fopenmp fhostone.F90 -O3 -o f.impi
	rm -f getcore.o

c.impi: phostone.c
	$(CC) $(OPS) -fopenmp phostone.c -O3 -o c.impi

pp.impi: ppong.c
	$(CC) $(OPS) $(WES) ppong.c -O3 -o pp.impi

clean:
	rm -rf *o *mod* f.impi c.impi pp.impi
```

- Other makefiles
  - Load different modules
  - Different F90, CC
  - Executable name change
- Makefile - do them all

# Setup and "make"

```
#!/usr/bin/bash
#SBATCH --job-name="affinity"
#SBATCH --nodes=2
#SBATCH --exclusive
#SBATCH --export=ALL
#SBATCH --time=04:00:00
#SBATCH --partition=standard

BASE=`pwd`

#Make a new directory and go there
STDIR=`pwd`
mkdir $SLURM_JOB_ID
cd $SLURM_JOB_ID

#optionally wait between launches
mywait () { sleep 0; }

#Copy everything
printenv > env
cat $0 > script

cp $BASE/make* .
cp $BASE/Makefile .
cp $BASE/fhostone.F90 .
cp $BASE/phostone.c .
cp $BASE/cases .
cp $BASE/post .
cp $BASE/ppong.c .
cp $BASE/getcore.c .
cp $BASE/maskgenerator.py .
cp $BASE/todo.py .
cp $BASE/tymer  .

tar -czf recreate.tgz *

#Create input for ppong
./todo.py

#Build our programs
make all > make.log 2>&1
make pp  > make.pp 2>&1

#Command line arguments for phostone
CLA="-i -F -E -t 7"
export FEXE=f
export CEXE=c
```

| | |
|---|---|
| make* | Makefiles |
| Makefile | Driver Makefile |
| cases | File of tasks and threads |
| post | Post processing script |
| maskgenerator.py | Manually creates mapping of threads to cores |
| todo.py | Creates input file for ppong |
| tymer | Nice wall clock  timer |

# Makefile (full)

```
all : impi  cray  gnu  intel open mpich openg mpichg dmod

impi: makeimpi
	make -f makeimpi

cray: makeprgcray
	make -f makeprgcray

gnu: makeprggnu
	make -f makeprggnu

intel: makeprgintel
	make -f makeprgintel

open: makeopen
	make -f makeopen

mpich: makempich
	make -f makempich


openg: makeopen_g
	make -f makeopen_g

mpichg: makempich_g
	make -f makempich_g

clean:
	make -f makeimpi clean
	make -f makeprgintel clean
	make -f makeprggnu clean
	make -f makeprgcray clean
	make -f makeopen clean
	make -f makempich clean
	make -f makeopen_g clean
	make -f makempich_g clean
	rm -rf runall.tgz simple.tgz
```

```
dmod:
	rm -rf *.o *mod

pp: pp.impi pp.cray pp.gnu pp.intel pp.open pp.oneapi pp.mpich pp.openg pp.mpichg

pp.impi: makeimpi
	make -f makeimpi pp.impi

pp.cray: makeprgcray
	make -f makeprgcray pp.cray

pp.gnu: makeprggnu
	make -f makeprggnu pp.gnu

pp.intel: makeprgintel
	make -f makeprgintel pp.intel

pp.open: makeopen
	make -f makeopen pp.open

pp.mpich: makempich
	make -f makempich pp.mpich


pp.openg: makeopen_g
	make -f makeopen_g pp.openg

pp.mpichg: makempich_g
	make -f makempich_g pp.mpichg

tar:
	tar -czf runall.tgz \
		cases eagle ecases fhostone.F90 getcore.c make1api Makefile makefile.include \
		makeimpi makeopen makeprgcray makeprggnu makeprgintel maskgenerator.py masks.txt \
		phostone.c post ppong.c readme.md runall runpp subsweep sweep todo.py tymer \
		scases array mapping.py simple makempich makempich_g makeopen_g

simple.tgz:
	tar -czf simple.tgz fhostone.F90  getcore.c  make1api  Makefile  makefile.include  \
		makefile.org  makeimpi  makeopen  makeprgcray  makeprggnu  makeprgintel \
		phostone.c  post  ppong.c simple makempich makempich_g makeopen_g
```

```
#LOOPING
export CRAY_OMP_CHECK_AFFINITY=TRUE
export nc=`cat cases | wc -l`
for il in `seq $nc` ; do
    aline=`cat cases | head -$il | tail -1`
    ntpn=`echo $aline | awk {'print $1'}`
    nthrd=`echo $aline | awk {'print $2'}`
    export OMP_NUM_THREADS=$nthrd
    for bindit in EMPTY SPREAD THREAD WORKS MASK ; do
        #export KMP_AFFINITY=scatter
        export OMP_PROC_BIND=spread
        export BIND=--cpu-bind=v,${bindit}
        unset CPUS_TASK
        if [ $bindit == MASK ] ; then
        cores=`expr $ntpn \* $nthrd`
        MASK=`./maskgenerator.py $cores $ntpn`
        BIND="--cpu-bind=v,mask_cpu:$MASK"
        fi
        if [ $bindit == NONE ] ; then
        BIND="--cpu-bind=v"
          export CPUS_TASK="--cpus-per-task=$nthrd"
        fi
...
...
        echo $ntpn $nthrd >> srunsettings
        echo $BIND $CPUS_TASK >> srunsettings
        printenv | egrep "OMP_|KMP_" >> srunsettings
        echo --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK >> srunsettings
...
...
```

# Looping

- Get a task/thread count from each line of cases
- We try various types of thread binding, spread (NONE) and manually (MASK)

- The script maskgenerator.py creates a string describing a mapping of tasks/threads to cores
- This is passed to run using the --cpu-bind option

- Save information for each iteration

```
./tymer mytimes PrgEnv-intel
          module purge
          module load craype-x86-spr
          module load intel
          module load PrgEnv-intel

./tymer mytimes fortran
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$FEXE.intel $CLA > f.intel.out_${ntpn}_${nthrd}_${bindit} \
                 2> f.intel.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes c
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$CEXE.intel $CLA > c.intel.out_${ntpn}_${nthrd}_${bindit} \
                 2> c.intel.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished

          if [[ $nthrd -eq 1 &&  $ntpn -eq 104 && $bindit == NONE ]] ; then
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./pp.intel $CLA > pp.intel.xxx_${ntpn}_${nthrd}_${bindit} \
                                   2> pp.intel.iii_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished ppong
          fi

./tymer mytimes PrgEnv-gnu
          module purge
          module load craype-x86-spr
          module load PrgEnv-gnu

./tymer mytimes fortran
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$FEXE.gnu $CLA > f.gnu.out_${ntpn}_${nthrd}_${bindit} \
                 2> f.gnu.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes c
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$CEXE.gnu $CLA > c.gnu.out_${ntpn}_${nthrd}_${bindit} \
                 2> c.gnu.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished

          if [[ $nthrd -eq 1 &&  $ntpn -eq 104 && $bindit == NONE ]] ; then
          mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./pp.gnu $CLA > pp.gnu.xxx_${ntpn}_${nthrd}_${bindit} \
                                   2> pp.gnu.iii_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished ppong
          fi
```

```
./tymer mytimes PrgEnv-cray
    module purge
    module load craype-x86-spr
    module load PrgEnv-cray

./tymer mytimes fortran
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$FEXE.cray $CLA > f.cray.out_${ntpn}_${nthrd}_${bindit} \
        2> f.cray.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes c
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$CEXE.cray $CLA > c.cray.out_${ntpn}_${nthrd}_${bindit} \
        2> c.cray.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished

    if [[ $nthrd -eq 1 &&  $ntpn -eq 104 && $bindit == NONE ]] ; then
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./pp.cray $CLA > pp.cray.xxx_${ntpn}_${nthrd}_${bindit} \
              2> pp.cray.iii_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished ppong
    fi

./tymer mytimes intel-oneapi
    module purge
    module load intel-oneapi
    module load libfabric

./tymer mytimes fortran
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$FEXE.impi $CLA > f.impi.out_${ntpn}_${nthrd}_${bindit} \
        2> f.impi.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes c
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./$CEXE.impi $CLA > c.impi.out_${ntpn}_${nthrd}_${bindit} \
        2> c.impi.info_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished

    if [[ $nthrd -eq 1 &&  $ntpn -eq 104 && $bindit == NONE ]] ; then
    mywait; srun --mpi=pmi2 $BIND  --tasks-per-node=$ntpn $CPUS_TASK  ./pp.impi $CLA > pp.impi.xxx_${ntpn}_${nthrd}_${bindit} \
              2> pp.impi.iii_${ntpn}_${nthrd}_${bindit}
./tymer mytimes finished ppong
    fi
    done
done
```

# Post processing

- Report of successful and failed phostone runs
- Copy slurm output to our final directory
- Might want to look at output from ppong
  - Bandwidth
  - MPI_Barrier rate
- Might want to look at MPI_Init times from phostone

```
./post
. ./post | sort -n > posit
getstate postit nope > report
getstate postit worked >> report
mv $STDIR/slurm-$SLURM_JOB_ID.out .
```

# REPORT

```
tkaiser2-37907s:177402 tkaiser2$ cat report
c.cray.out_104_1_MASK 208 208
c.cray.out_104_1_NONE 208 208
c.cray.out_1_104_MASK 208 208
c.cray.out_1_104_NONE 208 208
c.cray.out_1_8_MASK 16 16
c.cray.out_1_8_NONE 16 16
c.gnu.out_104_1_MASK 208 208
c.gnu.out_104_1_NONE 208 208
c.gnu.out_1_104_MASK 208 208
c.gnu.out_1_104_NONE 208 208
c.gnu.out_1_8_MASK 16 16
c.gnu.out_1_8_NONE 16 16
c.impi.out_104_1_MASK 208 208
c.impi.out_104_1_NONE 208 208
c.impi.out_1_104_MASK 208 208
c.impi.out_1_104_NONE 208 208
c.impi.out_1_8_MASK 16 16
c.impi.out_1_8_NONE 16 16
c.intel.out_104_1_MASK 208 208
c.intel.out_104_1_NONE 208 208
c.intel.out_1_104_MASK 208 208
c.intel.out_1_104_NONE 208 208
c.intel.out_1_8_MASK 16 16
c.intel.out_1_8_NONE 16 16
```

```
f.cray.out_104_1_MASK 208 208
f.cray.out_104_1_NONE 208 208
f.cray.out_1_104_MASK 208 208
f.cray.out_1_104_NONE 208 208
f.cray.out_1_8_MASK 16 16
f.cray.out_1_8_NONE 16 16
f.gnu.out_104_1_MASK 208 208
f.gnu.out_104_1_NONE 208 208
f.gnu.out_1_104_MASK 208 208
f.gnu.out_1_104_NONE 208 208
f.gnu.out_1_8_MASK 16 16
f.gnu.out_1_8_NONE 16 16
f.impi.out_104_1_MASK 208 208
f.impi.out_104_1_NONE 208 208
f.impi.out_1_104_MASK 208 208
f.impi.out_1_104_NONE 208 208
f.impi.out_1_8_MASK 16 16
f.impi.out_1_8_NONE 16 16
f.intel.out_104_1_MASK 208 208
f.intel.out_104_1_NONE 208 208
f.intel.out_1_104_MASK 208 208
f.intel.out_1_104_NONE 208 208
f.intel.out_1_8_MASK 16 16
f.intel.out_1_8_NONE 16 16
```

# RUN SETTINGS

AS NAMED IN THE SCRIPT

**EMPTY**

- export OMP_NUM_THREADS=$nthrd
- --cpu-bind=v

**SPREAD**

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v

**THREAD**

- export OMP_NUM_THREADS=$nthrd
- --cpu-bind=v
- --cpus-per-task=$nthrd

**WORKS**

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v,mask_cpu:$MASK
- --cpus-per-task=$nthrd

**MASK**

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v,mask_cpu:$MASK

## EMPTY

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 104 | 1 | 208 | 208 | |
| C,gnu | Cray | 104 | 1 | 208 | 208 | |
| C,Intel | INTEL | 104 | 1 | 208 | 208 | |
| C,Intel | Cray | 104 | 1 | 208 | 208 | |
| Fortran,Cray | Cray | 104 | 1 | 208 | 208 | |
| Fortran,gnu | Cray | 104 | 1 | 208 | 208 | |
| ifort | INTEL | 104 | 1 | 208 | 208 | |
| ifort | Cray | 104 | 1 | 208 | 208 | |
| C,Cray | Cray | 1 | 104 | 208 | 208 | |
| C,gnu | Cray | 1 | 104 | 208 | 208 | |
| C,Intel | INTEL | 1 | 104 | 208 | 153 | Failed |
| C,Intel | Cray | 1 | 104 | 208 | 152 | Failed |
| Fortran,Cray | Cray | 1 | 104 | 208 | 208 | |
| Fortran,gnu | Cray | 1 | 104 | 208 | 208 | |
| ifort | INTEL | 1 | 104 | 208 | 103 | Failed |
| ifort | Cray | 1 | 104 | 208 | 96 | Failed |
| C,Cray | Cray | 13 | 8 | 208 | 208 | |
| C,gnu | Cray | 13 | 8 | 208 | 208 | |
| C,Intel | INTEL | 13 | 8 | 208 | 207 | Failed |
| C,Intel | Cray | 13 | 8 | 208 | 208 | |
| Fortran,Cray | Cray | 13 | 8 | 208 | 106 | Failed |
| Fortran,gnu | Cray | 13 | 8 | 208 | 107 | Failed |
| ifort | INTEL | 13 | 8 | 208 | 143 | Failed |
| ifort | Cray | 13 | 8 | 208 | 137 | Failed |

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 1 | 8 | 16 | 16 | |
| C,gnu | Cray | 1 | 8 | 16 | 16 | |
| C,Intel | INTEL | 1 | 8 | 16 | 16 | |
| C,Intel | Cray | 1 | 8 | 16 | 16 | |
| Fortran,Cray | Cray | 1 | 8 | 16 | 16 | |
| Fortran,gnu | Cray | 1 | 8 | 16 | 16 | |
| ifort | INTEL | 1 | 8 | 16 | 16 | |
| ifort | Cray | 1 | 8 | 16 | 16 | |
| C,Cray | Cray | 8 | 13 | 208 | 208 | |
| C,gnu | Cray | 8 | 13 | 208 | 208 | |
| C,Intel | INTEL | 8 | 13 | 208 | 208 | |
| C,Intel | Cray | 8 | 13 | 208 | 208 | |
| Fortran,Cray | Cray | 8 | 13 | 208 | 111 | Failed |
| Fortran,gnu | Cray | 8 | 13 | 208 | 107 | Failed |
| ifort | INTEL | 8 | 13 | 208 | 150 | Failed |
| ifort | Cray | 8 | 13 | 208 | 164 | Failed |

- export OMP_NUM_THREADS=$nthrd
- --cpu-bind=v

## SPREAD

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 104 | 1 | 208 | 208 | |
| C,gnu | Cray | 104 | 1 | 208 | 208 | |
| C,Intel | INTEL | 104 | 1 | 208 | 208 | |
| C,Intel | Cray | 104 | 1 | 208 | 208 | |
| Fortran,Cray | Cray | 104 | 1 | 208 | 208 | |
| Fortran,gnu | Cray | 104 | 1 | 208 | 208 | |
| ifort | INTEL | 104 | 1 | 208 | 208 | |
| ifort | Cray | 104 | 1 | 208 | 208 | |
| C,Cray | Cray | 1 | 104 | 208 | 208 | |
| C,gnu | Cray | 1 | 104 | 208 | 208 | |
| C,Intel | INTEL | 1 | 104 | 208 | 208 | |
| C,Intel | Cray | 1 | 104 | 208 | 208 | |
| Fortran,Cray | Cray | 1 | 104 | 208 | 208 | |
| Fortran,gnu | Cray | 1 | 104 | 208 | 208 | |
| ifort | INTEL | 1 | 104 | 208 | 208 | |
| ifort | Cray | 1 | 104 | 208 | 208 | |
| C,Cray | Cray | 13 | 8 | 208 | 16 | Failed |
| C,gnu | Cray | 13 | 8 | 208 | 16 | Failed |
| C,Intel | INTEL | 13 | 8 | 208 | 16 | Failed |
| C,Intel | Cray | 13 | 8 | 208 | 16 | Failed |
| Fortran,Cray | Cray | 13 | 8 | 208 | 16 | Failed |
| Fortran,gnu | Cray | 13 | 8 | 208 | 16 | Failed |
| ifort | INTEL | 13 | 8 | 208 | 16 | Failed |
| ifort | Cray | 13 | 8 | 208 | 16 | Failed |

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 1 | 8 | 16 | 16 | |
| C,gnu | Cray | 1 | 8 | 16 | 16 | |
| C,Intel | INTEL | 1 | 8 | 16 | 16 | |
| C,Intel | Cray | 1 | 8 | 16 | 16 | |
| Fortran,Cray | Cray | 1 | 8 | 16 | 16 | |
| Fortran,gnu | Cray | 1 | 8 | 16 | 16 | |
| ifort | INTEL | 1 | 8 | 16 | 16 | |
| ifort | Cray | 1 | 8 | 16 | 16 | |
| C,Cray | Cray | 8 | 13 | 208 | 26 | Failed |
| C,gnu | Cray | 8 | 13 | 208 | 26 | Failed |
| C,Intel | INTEL | 8 | 13 | 208 | 26 | Failed |
| C,Intel | Cray | 8 | 13 | 208 | 26 | Failed |
| Fortran,Cray | Cray | 8 | 13 | 208 | 26 | Failed |
| Fortran,gnu | Cray | 8 | 13 | 208 | 26 | Failed |
| ifort | INTEL | 8 | 13 | 208 | 26 | Failed |
| ifort | Cray | 8 | 13 | 208 | 26 | Failed |

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v

NREL
Transforming ENERGY

## THREAD

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 104 | 1 | 208 | 208 | |
| C,gnu | Cray | 104 | 1 | 208 | 208 | |
| C,Intel | INTEL | 104 | 1 | 208 | 208 | |
| C,Intel | Cray | 104 | 1 | 208 | 208 | |
| Fortran,Cray | Cray | 104 | 1 | 208 | 208 | |
| Fortran,gnu | Cray | 104 | 1 | 208 | 208 | |
| ifort | INTEL | 104 | 1 | 208 | 208 | |
| ifort | Cray | 104 | 1 | 208 | 208 | |
| C,Cray | Cray | 1 | 104 | 208 | 208 | |
| C,gnu | Cray | 1 | 104 | 208 | 208 | |
| C,Intel | INTEL | 1 | 104 | 208 | 155 | |
| C,Intel | Cray | 1 | 104 | 208 | 167 | |
| Fortran,Cray | Cray | 1 | 104 | 208 | 208 | |
| Fortran,gnu | Cray | 1 | 104 | 208 | 208 | |
| ifort | INTEL | 1 | 104 | 208 | 98 | Failed |
| ifort | Cray | 1 | 104 | 208 | 96 | Failed |
| C,Cray | Cray | 13 | 8 | 208 | 208 | |
| C,gnu | Cray | 13 | 8 | 208 | 208 | |
| C,Intel | INTEL | 13 | 8 | 208 | 206 | Failed |
| C,Intel | Cray | 13 | 8 | 208 | 205 | Failed |
| Fortran,Cray | Cray | 13 | 8 | 208 | 208 | |
| Fortran,gnu | Cray | 13 | 8 | 208 | 208 | |
| ifort | INTEL | 13 | 8 | 208 | 155 | Failed |
| ifort | Cray | 13 | 8 | 208 | 155 | Failed |

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 1 | 8 | 16 | 16 | |
| C,gnu | Cray | 1 | 8 | 16 | 16 | |
| C,Intel | INTEL | 1 | 8 | 16 | 16 | |
| C,Intel | Cray | 1 | 8 | 16 | 16 | |
| Fortran,Cray | Cray | 1 | 8 | 16 | 16 | |
| Fortran,gnu | Cray | 1 | 8 | 16 | 16 | |
| ifort | INTEL | 1 | 8 | 16 | 16 | |
| ifort | Cray | 1 | 8 | 16 | 16 | |
| C,Cray | Cray | 8 | 13 | 208 | 208 | |
| C,gnu | Cray | 8 | 13 | 208 | 208 | |
| C,Intel | INTEL | 8 | 13 | 208 | 208 | |
| C,Intel | Cray | 8 | 13 | 208 | 208 | |
| Fortran,Cray | Cray | 8 | 13 | 208 | 208 | |
| Fortran,gnu | Cray | 8 | 13 | 208 | 208 | |
| ifort | INTEL | 8 | 13 | 208 | 208 | |
| ifort | Cray | 8 | 13 | 208 | 208 | |

- export OMP_NUM_THREADS=$nthrd
- --cpu-bind=v
- --cpus-per-task=$nthrd
- *--threads-per-core=1*

NREL Transforming ENERGY

## WORKS

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 104 | 1 | 208 | 208 | |
| C,gnu | Cray | 104 | 1 | 208 | 208 | |
| C,Intel | INTEL | 104 | 1 | 208 | 208 | |
| C,Intel | Cray | 104 | 1 | 208 | 208 | |
| Fortran,Cray | Cray | 104 | 1 | 208 | 208 | |
| Fortran,gnu | Cray | 104 | 1 | 208 | 208 | |
| ifort | INTEL | 104 | 1 | 208 | 208 | |
| ifort | Cray | 104 | 1 | 208 | 208 | |
| C,Cray | Cray | 1 | 104 | 208 | 208 | |
| C,gnu | Cray | 1 | 104 | 208 | 208 | |
| C,Intel | INTEL | 1 | 104 | 208 | 208 | |
| C,Intel | Cray | 1 | 104 | 208 | 208 | |
| Fortran,Cray | Cray | 1 | 104 | 208 | 208 | |
| Fortran,gnu | Cray | 1 | 104 | 208 | 208 | |
| ifort | INTEL | 1 | 104 | 208 | 208 | |
| ifort | Cray | 1 | 104 | 208 | 208 | |
| C,Cray | Cray | 13 | 8 | 208 | 208 | |
| C,gnu | Cray | 13 | 8 | 208 | 208 | |
| C,Intel | INTEL | 13 | 8 | 208 | 208 | |
| C,Intel | Cray | 13 | 8 | 208 | 208 | |
| Fortran,Cray | Cray | 13 | 8 | 208 | 208 | |
| Fortran,gnu | Cray | 13 | 8 | 208 | 208 | |
| ifort | INTEL | 13 | 8 | 208 | 208 | |
| ifort | Cray | 13 | 8 | 208 | 208 | |

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 1 | 8 | 16 | 16 | |
| C,gnu | Cray | 1 | 8 | 16 | 16 | |
| C,Intel | INTEL | 1 | 8 | 16 | 16 | |
| C,Intel | Cray | 1 | 8 | 16 | 16 | |
| Fortran,Cray | Cray | 1 | 8 | 16 | 16 | |
| Fortran,gnu | Cray | 1 | 8 | 16 | 16 | |
| ifort | INTEL | 1 | 8 | 16 | 16 | |
| ifort | Cray | 1 | 8 | 16 | 16 | |
| C,Cray | Cray | 8 | 13 | 208 | 208 | |
| C,gnu | Cray | 8 | 13 | 208 | 208 | |
| C,Intel | INTEL | 8 | 13 | 208 | 208 | |
| C,Intel | Cray | 8 | 13 | 208 | 208 | |
| Fortran,Cray | Cray | 8 | 13 | 208 | 208 | |
| Fortran,gnu | Cray | 8 | 13 | 208 | 208 | |
| ifort | INTEL | 8 | 13 | 208 | 208 | |
| ifort | Cray | 8 | 13 | 208 | 208 | |

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v
- --cpus-per-task=$nthrd

## MASK

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 104 | 1 | 208 | 208 | |
| C,gnu | Cray | 104 | 1 | 208 | 208 | |
| C,Intel | INTEL | 104 | 1 | 208 | 208 | |
| C,Intel | Cray | 104 | 1 | 208 | 208 | |
| Fortran,Cray | Cray | 104 | 1 | 208 | 208 | |
| Fortran,gnu | Cray | 104 | 1 | 208 | 208 | |
| ifort | INTEL | 104 | 1 | 208 | 208 | |
| ifort | Cray | 104 | 1 | 208 | 208 | |
| C,Cray | Cray | 1 | 104 | 208 | 208 | |
| C,gnu | Cray | 1 | 104 | 208 | 208 | |
| C,Intel | INTEL | 1 | 104 | 208 | 208 | |
| C,Intel | Cray | 1 | 104 | 208 | 208 | |
| Fortran,Cray | Cray | 1 | 104 | 208 | 208 | |
| Fortran,gnu | Cray | 1 | 104 | 208 | 208 | |
| ifort | INTEL | 1 | 104 | 208 | 208 | |
| ifort | Cray | 1 | 104 | 208 | 208 | |
| C,Cray | Cray | 13 | 8 | 208 | 208 | |
| C,gnu | Cray | 13 | 8 | 208 | 208 | |
| C,Intel | INTEL | 13 | 8 | 208 | 208 | |
| C,Intel | Cray | 13 | 8 | 208 | 208 | |
| Fortran,Cray | Cray | 13 | 8 | 208 | 208 | |
| Fortran,gnu | Cray | 13 | 8 | 208 | 208 | |
| ifort | INTEL | 13 | 8 | 208 | 208 | |
| ifort | Cray | 13 | 8 | 208 | 208 | |

WORKS-1

| Language | MPI | Tasks | Threads | Expected Cores | Cores Used | Status |
|---|---|---|---|---|---|---|
| C,Cray | Cray | 1 | 8 | 16 | 16 | |
| C,gnu | Cray | 1 | 8 | 16 | 16 | |
| C,Intel | INTEL | 1 | 8 | 16 | 16 | |
| C,Intel | Cray | 1 | 8 | 16 | 16 | |
| Fortran,Cray | Cray | 1 | 8 | 16 | 16 | |
| Fortran,gnu | Cray | 1 | 8 | 16 | 16 | |
| ifort | INTEL | 1 | 8 | 16 | 16 | |
| ifort | Cray | 1 | 8 | 16 | 16 | |
| C,Cray | Cray | 8 | 13 | 208 | 208 | |
| C,gnu | Cray | 8 | 13 | 208 | 208 | |
| C,Intel | INTEL | 8 | 13 | 208 | 208 | |
| C,Intel | Cray | 8 | 13 | 208 | 208 | |
| Fortran,Cray | Cray | 8 | 13 | 208 | 208 | |
| Fortran,gnu | Cray | 8 | 13 | 208 | 208 | |
| ifort | INTEL | 8 | 13 | 208 | 208 | |
| ifort | Cray | 8 | 13 | 208 | 208 | |

- export OMP_NUM_THREADS=$nthrd
- export OMP_PROC_BIND=spread
- --cpu-bind=v,mask_cpu:$MASK

# ABOUT MASKS

- Masks are specified as a string with N task values
- Each value gives the cores on which MPI task N-1 can reside
- Masks are either Decimal or Hex but are interpreted as binary
- Does not guarantee threads will be even distributed between allowed cores

```
mask=0xfff80000000000000000000000,0x7ffc0000000000000000000,0x3ffe0000000000000000,0x1fff0000000000000,0xfff8000000000,
0x7ffc000000,0x3ffe000,0x1fff
>>> for task in range(0,8) :
...     print(task, f'{input[task]:0104b}')
...
0 11111111111110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
1 00000000000001111111111111000000000000000000000000000000000000000000000000000000000000000000000000000000
2 00000000000000000000000000111111111111100000000000000000000000000000000000000000000000000000000000000000
3 00000000000000000000000000000000000000011111111111110000000000000000000000000000000000000000000000000000
4 00000000000000000000000000000000000000000000000000001111111111111000000000000000000000000000000000000000
5 00000000000000000000000000000000000000000000000000000000000000000111111111111100000000000000000000000000
6 00000000000000000000000000000000000000000000000000000000000000000000000000000000011111111111110000000000000
7 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111111
>>>
```

8 Tasks, 13 threads

# RECOMMENDATIONS

- Download the repo
  - Change module loads to match you system
  - Run it
- Build phostone with your environment
- Run with your srun and env settings to see that you have proper mappings
  - May want to add --threads-per-core=1 to your srun line
- **Don't rely on "default" settings and assume you have good affinity**

# REPO AND CONTACT

https://github.com/timkphd/affinity.git

tkaiser2@nrel.gov